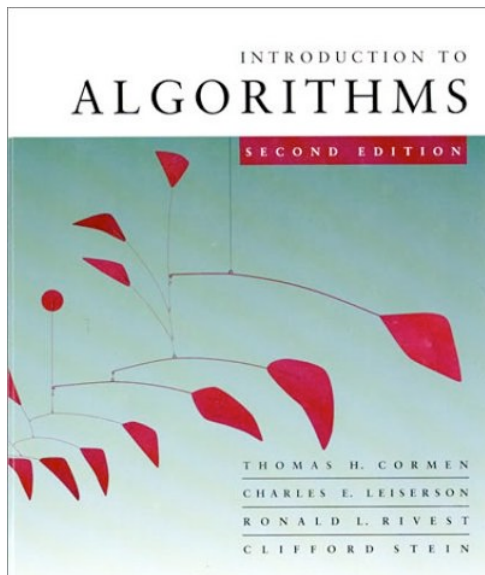


Algorithms and Data Structures

CSE 465

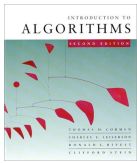


LECTURE 23

Augmenting Data Structures

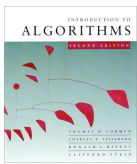
- Dynamic order statistics
- Methodology
- Interval trees

S. Raskhodnikova and A. Smith



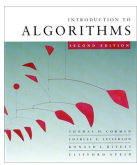
Greedy Algorithms

- Build-up a solution to an optimization problem, short-sightedly choosing the best option at each step
 - ♦ Sometimes good (often not!)
- Strategies for proving the correctness of a greedy algorithm
 - ♦ “Greedy Stays Ahead”: prove that at each step, greedy strategy does as well as optimal (last lecture, “interval scheduling”)
 - ♦ “Exchange argument”: show that any solution can be improved by swapping two choices to make it look more like the greedy solution (today)



Maximum Lateness

- Given: a list of n jobs with
 - ♦ duration t
 - ♦ deadline d (assume distinct)
- We want to find a good schedule for executing the jobs on a single machine, starting at time 0
 - ♦ Output: $s[1 .. n], f[1 .. n]$
 - ♦ Want: each job completed before its deadline (may not be possible)
 - ♦ Lateness of a job: (completion time) - deadline
 - ♦ **Goal:** make maximum lateness as small as possible



Greedy Strategies?

- Candidates:
 - ♦ in order of duration, shortest first?
 - ♦ in order of slack (deadline - duration), least slack first?
 - ♦ in order of deadline, soonest first?
- Quantity being optimized matters a lot
 - ♦ Different criteria \Rightarrow different algorithms

Scheduling by soonest deadline

- Optimal algorithm:
 - ♦ sort jobs by deadline $O(n \log n)$
 - ♦ run in that order with no idle time
 - Suppose the deadlines are $d[1] \leq d[2] \leq \dots \leq d[n]$
 - Output $s[1]=0, f[1] = t[1], s[i] = f[i-1], f[i] = s[i] + t[i]$
- Why is this the best possible?
 - ♦ Start from any valid solution, make it look more like greedy solution and show that it only makes the max. lateness smaller.
- Step 1: there is an optimal solution with no idle time
 - ♦ eliminating idle time only reduces completion times

Cont'd

- Now, given some schedule, we say there is an **inversion** if there are two jobs i, j where i is scheduled before j , but $d[i] > d[j]$
- Step 2: Any two solutions with no idle time and no inversions have the same max. lateness
 - ♦ Proof: if two different schedules have no idle time and no inversions, they differ only in the order in which jobs with identical deadlines are scheduled
 - ♦ Switching those jobs around doesn't change the time at which the last of them finishes

Cont'd

- Step 3: Exchange argument
 - ♦ Suppose there is a schedule which does have an inversion
 - ♦ (a) There is a *consecutive* inverted pair i, j
 - ♦ (b) Swap that pair
 - Only the finishing times of i and j are affected by the swap
 - If they are not the cause of maximum lateness, then nothing changes
 - Before swap, j has worse lateness
 - After swap, both are better than j was before
- Conclusion: there is an optimal schedule with no inversions and no idle time